SMP2 Modelling using the K2 Simulation infrastructure

Workshop on Simulation for European Space Programmes (SESP)
26 March 2015

ESA-ESTEC, Noordwijk, The Netherlands

Laurent Cohen[(1)], Anthony Mollier[(1)], Stéphane Vinay[(1)]

[(1)]Thales Alenia Space France
5 allées des Gabians
06150 Cannes
Email:laurent.cohen@thalesaleniaspace.com

ABSTRACT

The K2 Simulation platform is used for modelling and simulating test benches and satellites (a constellation of satellites, a single satellite, or a subset of a satellite). The runtime environment executes a simulation in real-time that can be monitored and controlled using customisable graphical user interfaces. The simulation data and results are stored in a database that can be processed afterwards.

The K2 simulation platform is currently improved to be fully compatible with the SMP2 standard.
The aim is to be able to import any models developed on the K2 platform into any SMP2 compatible environment. The compatibility is certified using the BASILES and SimSat frameworks. This adds the capability to share the K2 huge models database developed over the years.
On the other hand, it will be also possible to import any SMP2 compatible models into a K2 platform environment. The connections and communications between K2 and SMP2 models will be completely transparent. The scheduling of the models execution, the logger and many other services will be handled by the K2 platform.

The K2Lab framework delivered with the K2 simulation platform is a graphical application used to design models and simulators. It also generates the models interfaces, requirements and validation documentation templates.
The model designer tool enables the user to specify the models interfaces, algorithms and internal variables. The model developer adds only the behaviour to the automatically generated skeleton code, and can go back to the design without losing its manual additions.
The simulator designer tool is used to create instances of models and connect them together. A type check is performed, forcing the user to connect only compatible types.
The K2Lab framework will be improved to generate SMP2 compatible code at the same time as the K2 code. It will also contain a graphical user interface to launch, monitor and control the designed simulators for debug purposes.

INTRODUCTION

This paper describes the main characteristics of the K2 simulation platform, the models and the ways to connect them together in a simulator. The main part of the document describes how the existing K2 platform will evolve to be SMP2 compatible. The platform simulation tools to design K2/SMP2 models and execute the code are also presented in this document.

THE K2 SIMULATION PLATFORM

The K2 simulation platform is composed of a set of C++, Python and Java libraries and a collection of third party binaries. It is delivered under a Thales Alenia Space France proprietary commercial software licence.
The K2 models are developed in C++ using these libraries. They are instantiated and connected in simulators using a specific language based on python (all the existing python libraries and python commands are available to develop simulators). The validation tests of models and simulators are also prepared in python.
Swig and JNI are used to interconnect C++, Java and Python source code.

The K2 model structure is predefined and contains:

- A set of data defining the state of the model at a given time. These data can be accessed only from inside the model, they are not visible for other models:
  - Features : model static characteristics, do not change during run time
  - Shared Features : model static characteristics common to several models
  - States : values describing the current model state
  - Internal : internal model data
- A set of algorithms implemented by the model. A K2 algorithm is equivalent to an SMP2 Entry Point, it does not take any parameter and does not return a value:
  - An algorithm is a portion of code executed in the model
  - It can be executed once or periodically, with a priority
  - It can read the model inputs and data and set the outputs
  - Some special algorithms are specified and called by the platform, the others are user defined
- A set of model inputs:
  - A model input can be a value (simple K2 type mapped on C++ type or complex structures and arrays of these simple K2 types)
  - A routine is a function called by another model (or by the model itself)
    - It has parameters set by the caller
    - It returns a value to the caller
    - A bus routine uses addresses and masks to be executed
    - The equivalent concept in SMP2 is called “Operations”
- A set of model outputs:
  - The outputs are mirroring the inputs. An output can be connected to N inputs
  - The callpoint calls the routines. N callpoints can be connected to M routines
  - The connection between inputs and outputs can be seen as “pointers”. The data propagation (the input value is set to the corresponding output value straightaway) is a natural K2 feature. The same behaviour can be obtained using the MdkDP delivered with the Basiles SMP2 service

The following figure shows a graphical view of a K2 model content:

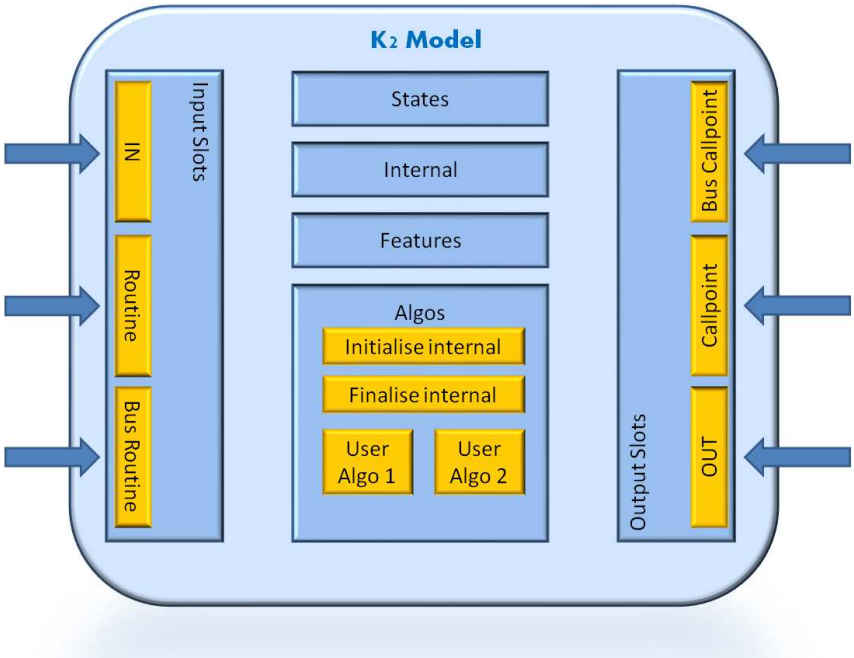

Fig. 1. K2 Model content

The K2 platform provides a group of services for the simulators. Here is a non-exhaustive list of available services, all of them are compliant with the SMP2 architecture:

- A scheduler
- A logger that can be filtered by criticality
- A time keeper providing different type of time: simulation time, mission time, zulu time
- A tracer triggered by a timer allowing to monitor the models data periodically
- An activation tracer triggered when a routine is called allowing to monitor the parameters and return values of each call

- An automatic save and restore feature

The scheduler manages a list of cyclic and acyclic events.
An event is defined by:

- An algorithm
- A date of first execution
- A priority
- An optional frequency of execution

The simulation time is stopped during the event execution (all the events are executed instantaneously). The events table is created at the beginning of the simulation and updated at run time: an algorithm or a routine can post events in the scheduler dynamically. The K2 platform offers different types of schedulers: rate monotonic, first posted, first executed, … The scheduler policy can be changed before the beginning of the simulation.

The cyclic events are posted only once in the scheduler. After the event execution, it is automatically re-posted for the next cycle. One (at most) of the simulator models can be the master of time of the simulation: it has to implement an algorithm called by the scheduler after each cycle of execution. This algorithm is responsible of shifting the simulation time. It is the only algorithm spending simulation time.

A K2 simulator is a self-contained application that runs without any user interaction. It generates logs and profiler information that can be analysed either dynamically during the simulation execution or after execution.

Nevertheless, the K2 Simulator can instantiate a XmlRpc server able to receive commands from an external tool. The simulation and models can be fully controlled from outside. Here is a non-exhaustive list of possibilities allowed by this embedded server : change the simulation execution speed, post new events to be executed in the scheduler, call algorithms, disconnect or reconnect input/outputs, pause and resume the simulation, dynamically add a new python function to the simulation and execute it, execute a scenario of actions (cross-check), introspect and modify all the models features, send a TC, monitor TMs, …

K2LAB

K2Lab is a graphical application developed on top of the eclipse RCP framework. It is based on a standard C++ eclipse with a set of plugins developed by Thales Alenia Space to design and execute K2 models and simulators.
One of the main benefit when developing with eclipse RCP is the modularity. Thales Alenia Space has developed a huge set of independent tools that can be put together to create an integrated application to design, execute, monitor, control and validate simulators.

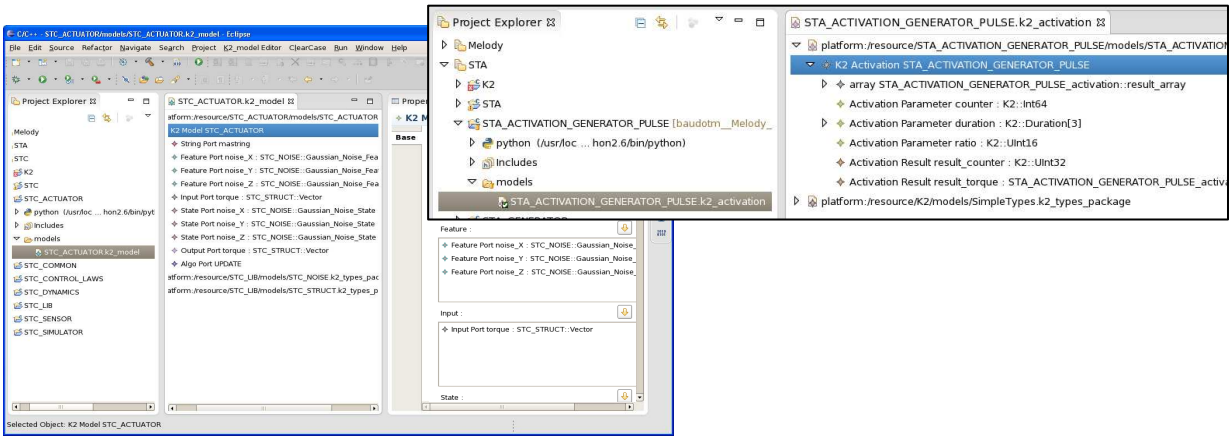

Fig. 2. K2Lab graphical application

The K2Lab modelling plugin is based on EMF (Eclipse Modelling Framework) and Acceleo. It contains the K2 “ecore” scheme and all the necessary K2 data types to design a K2 model. The developer can create its own data types libraries based on the basic K2 types and reuse them.

This modelling tool is managing the same sort of data than the SMP2 catalogue file. It allows the user to define the interfaces of the model, but also its internal data structure and the algorithms implemented by the model.
By using reverse engineering, K2Lab is able to read an existing code (K2 header files) and create the associated model.

The model created using K2Lab can easily be translated into an equivalent SMP2 catalogue file: all the necessary information to create the catalogue are available, assuming the K2 and SMP2 concepts for the interfaces are equivalent (this is described in the next paragraph).

SMP2 COMPATIBILITY FOR NATIVE K2 MODELS

It is assumed that the SMP2 specification is known by the reader (References [1] and [2]). The following paragraphs focus on how the SMP2 concepts are mapped into K2 concepts and conversely.

The K2 basic types are fully compatible with the SMP2 types. The connection concepts between models are however quite different and need a translation layer. This translation layer is automatically generated by wrapping the K2 code into an SMP2 model. The objective is to use the K2 model as a functional model and to wrap it around an SMP2 model knowing only the SMP2 interface of the model and how to connect it to the K2 functional model. The following figure illustrates how a K2 model becomes SMP2 compatible

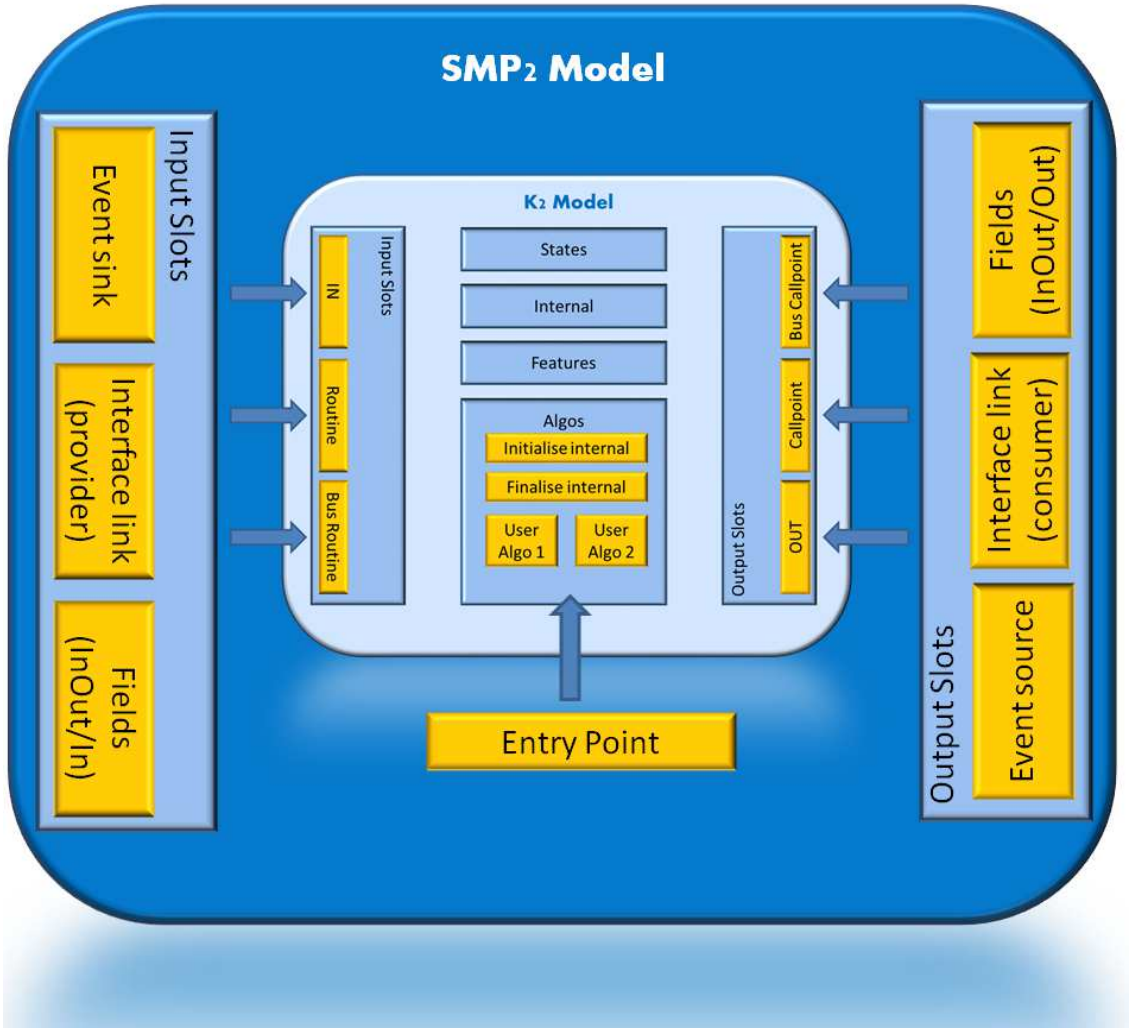

Fig. 3. K2 Model wrapped into an SMP2 shell

These operations can be fully automated if the translation of K2 concepts into SMP2 concepts can be automated. We have noticed that this is possible most of the time. Some exceptions are nevertheless possible when the SMP2 model specification enforce the use of a particular SMP2 type of connection or for performance reasons (An example of reason is the ISIS interface for the TM/TC service that forces the use of SMP2 “Interfaces”) . For these exceptions, K2Lab will provide a graphical interface to overwrite for a specific interface the default translation with the one chosen by the model developer before generating the code.

The wrapper code implements the connection between the K2 functional model and the SMP2 interface as shown in the following figure :

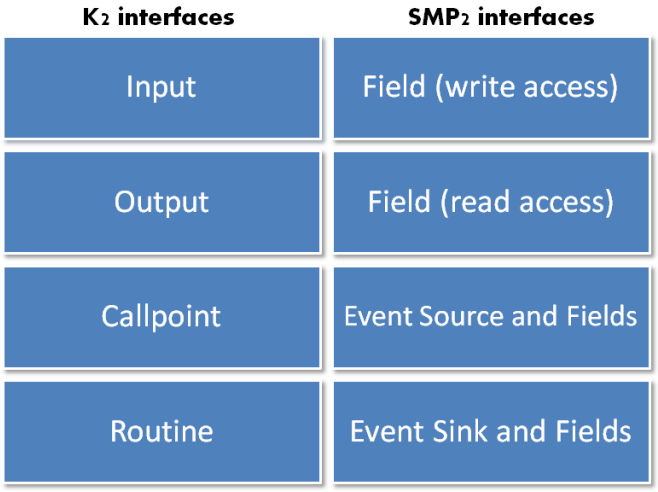

Fig. 4. K2/SMP2 interface translation

The K2 algorithms are defined as SMP2 entry points by the wrapper in order to use the SMP2 runtime scheduler instead of the K2 runtime scheduler. By contrast, it shall not be useful to access the States, Internal and Features data of the K2 model from outside, and therefore from the SMP2 shell. The default behaviour is to not translate these data as SMP2 design elements.

The following figure describes how K2Lab generates an SMP2 compatible model skeleton from the model design

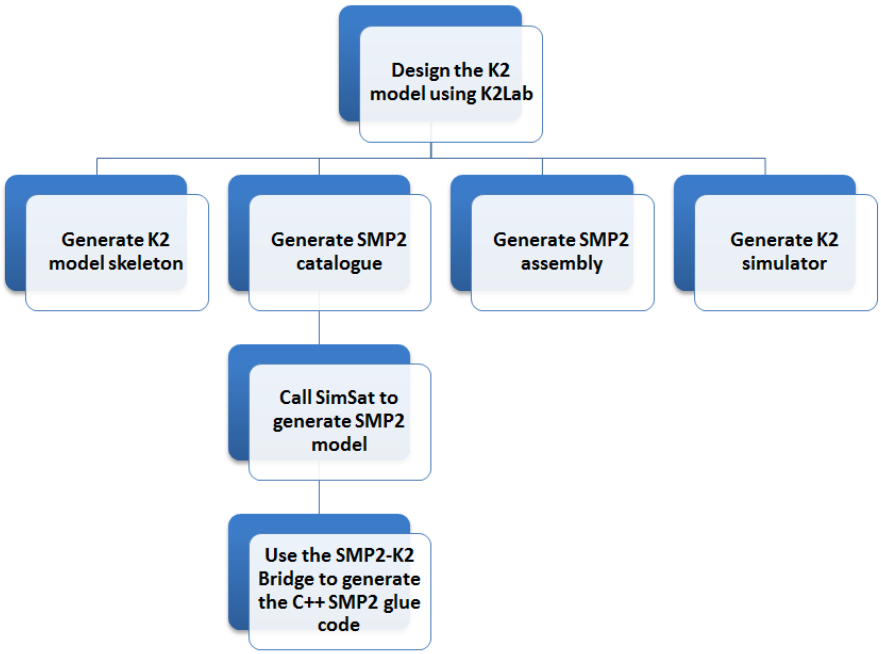

Fig. 5. Generate an SMP2 model using the K2Lab modelling tool

The generated SMP2 model embeds the K2 model and some necessary K2 platform libraries to compile correctly. It can be connected to other SMP2 models using an assembly and executed in an SMP2 environment as any SMP2 native model.

K2 COMPATIBILITY FOR NATIVE SMP2 MODELS

The K2 model generator uses the SMP2 catalogue file of the model as input. The same mechanisms are applied to automatically wrap the SMP2 model into a K2 shell. The aim of this procedure is to connect and execute an existing SMP2 model in a K2 environment running K2 models. The SMP2 embedded model becomes the functional model and the K2 model describes only the interface.

The following figure shows how an SMP2 model is encapsulated into a K2 model

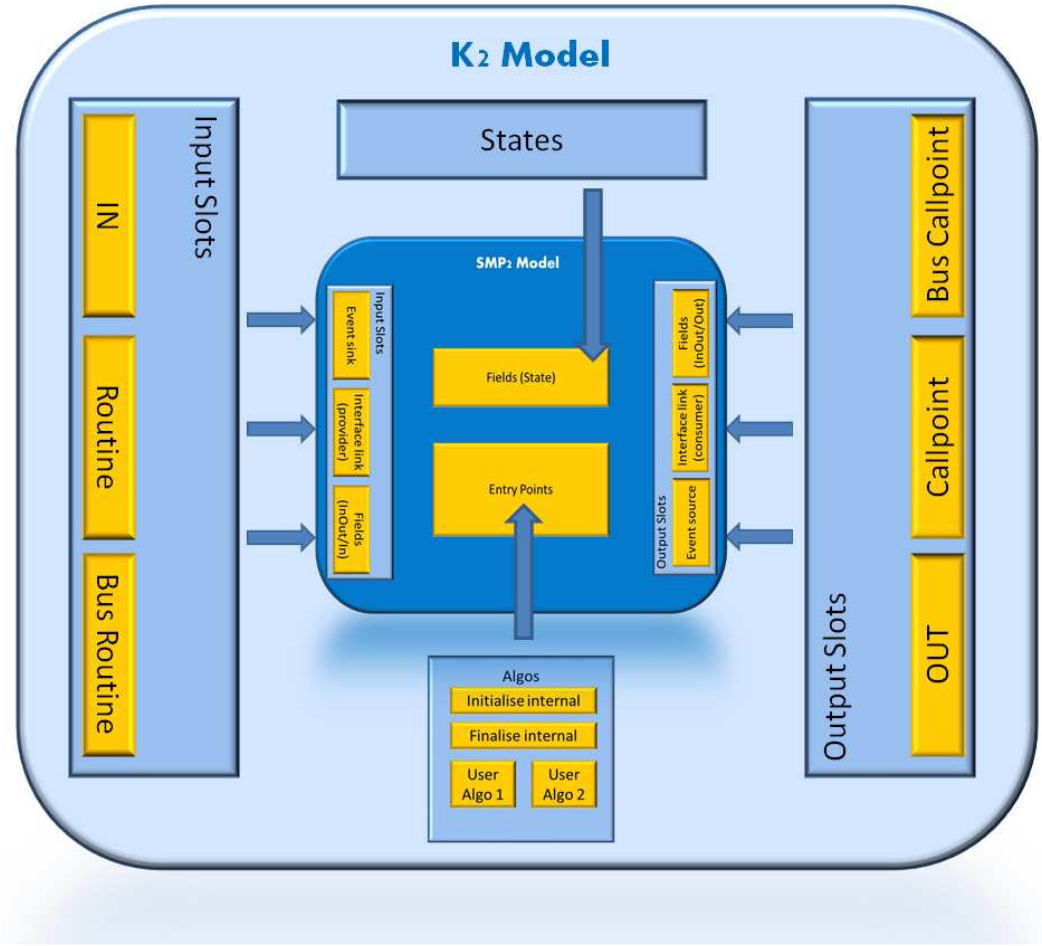

Fig. 6. SMP2 Model wrapped into a K2 shell

The following figure describes how K2Lab generates a K2 model skeleton from the SMP2 catalogue

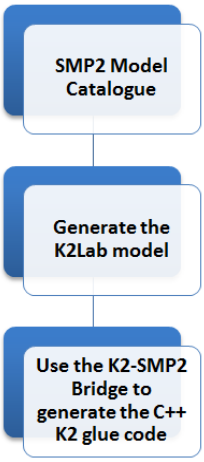

Fig. 7. Generate a K2 model using the K2Lab modelling tool with an SMP2 catalogue

The following figure shows the mapping between the SMP2 design elements and the K2 interfaces:

SMP2 design element	K2 interfaces
Input Field	Input
Output Field	Output
InOut Field	Input and Output
Field	State
Event Source	Callpoint
Event Sink	Routine
Interface	Callpoint/Routine

Fig. 8. Translation of SMP2 design elements in K2 interfaces

Most of the SMP2 elements are easily translated into K2 interfaces. The “InOut” fields translation is however more difficult: this concept does not exist in K2, therefore the wrapper needs to implement a mechanism to duplicate the “InOut” field into an Input field and an Output field, and copy the updated value from one to the other when needed.

The SMP2 Interface concept is a C++ native concept that could have been used as is when encapsulated in a K2 model: the K2 models are able to use the SMP2 interface as a standard C++ interface without any modification. There is however some limitations when using the SMP2 interface concepts: this way to exchange information between models is not using a native K2 concept. Therefore, the monitoring and control tools of the K2 platform are not aware of these interactions.
To resolve this problem, we chose to translate the SMP2 interfaces into K2 Callpoints and Routines. The wrapper of the models implementing the interface (producer) generates K2 Routines and the wrapper of the models using the interface (consumer) generates K2 Callpoints

K2/SMP2 MODELS VALIDATION

The SMP2 models validation tests using BASILES are prepared with the TCL language. On the other hand, the K2 models validation tests are prepared with the Python language. We did not find a way to easily convert the K2 Python tests into BASILES compatible tests. The selected solution is to validate the functional K2 model using the K2 platform and to validate the interfaces using the BASILES platform, without re-testing the functional behaviour and reciprocally for SMP2 models.

The functional validation is made in continuous integration using Jenkins and Sonar (Commercial Off-The-Shelf) to check that all the model functionality are implemented and running correctly during the development. We have also developed a set of eclipse RCP plugins (grouped in a framework application called eTestLab) to monitor and control the real-time execution of a simulator, and particularly of the models. These plugins will be integrated to K2Lab to have an integrated model development tool allowing to design, develop and validate models.

The following figures shows some of the eTestLab views.

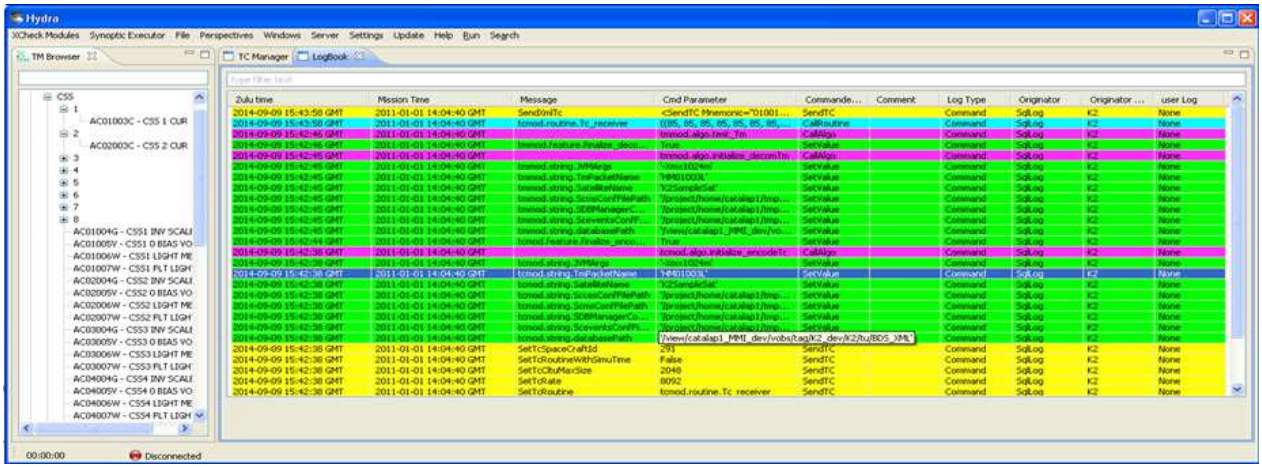

Fig. 9. Logbook view of eTestLab

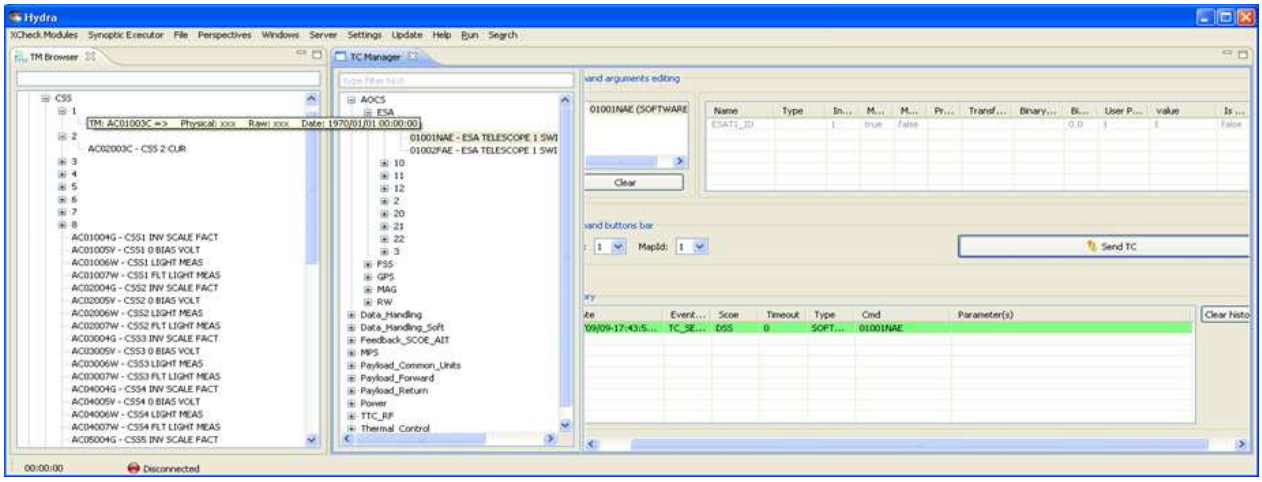

Fig. 10. TM/TC views of eTestLab

CONCLUSION

Thales Alenia Space has a huge experience using the K2 platform to create simulators. A large number of tools and models have been developed around this platform.
The K2 and SMP2 concepts are quite different in some aspects. Having a K2 platform natively compatible with the SMP2 standard would change some of the K2 foundations. On the other hands, the need to share our models with other companies and to integrate external models to our simulators pushed us to find a solution to be SMP2 compatible.
The solutions presented in this document will be fully automated and operational on Q3 2015. This will open up new opportunities to share and reuse existing models. However this approach is seen as the first step to have a K2 platform natively compatible with the SMP2 standard.

REFERENCES

[1] Peter Ellsiepen, Peter Fritzen, SMP 2.0 Handbook EGOS-SIM-GEN-TN-0099 Issue 1 Revision 2 28 October 2005
[2] Peter Fritzen, Stephan Kranz, Peter Ellsiepen SMP 2.0 C++ Mapping EGOS-SIM-GEN-TN-0102 Issue 1 Revision 2 28 October 2005